Algebra Polynom

Generated by Doxygen 1.9.4

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 GaloisField Class Reference

class GaloisField that describes finite field of polynomials generated by irreducible polynomial

`#include <GaloisField.h>`

Public Member Functions

- Polynom **add** (Polynom const &left, Polynom const &right) const
  Adds polynomials in the field.
- Polynom **subtract** (Polynom const &left, Polynom const &right) const
  Subtracts polynomials in the field.
- Polynom **multiply** (Polynom const &left, Polynom const &right) const
  Multiplies polynomials in the field.
- Polynom **extendedEuclideanAlgorithm** (Polynom a, Polynom b, Polynom ∗x, Polynom ∗y) const
  Implements extended Euclidean algorithm.
- Polynom **getInverse** (Polynom const &polynom) const
  Finds the inverse of a given polynomial in the field.
- Polynom **divide** (Polynom const &left, Polynom const &right) const
  Divides polynomials in the field.
- Polynom **mod** (Polynom const &left, Polynom const &right) const
  Takes the remainder of a division of polynomials in the field.
- Polynom **derivative** (Polynom const &polynom) const
  Finds the derivative of a given polynomial.

3.1.1 Detailed Description

class GaloisField that describes finite field of polynomials generated by irreducible polynomial

@autor Pashchenko Dmytro

defines a field of polynomials over the field GF(p) (p is prime) irreducible polynomial of power "degree"

The documentation for this class was generated from the following file:

- Algebra-polynom/Polynom/GaloisField.h

3.2 Polynom Class Reference

class Polynom that describes polynomial in a ring GF(p)[X]

`#include <Polynom.h>`

Public Member Functions

- std::string **show** () const

  Returns string representation of the polynomial.
- long long **getPolyPower** () const

  Returns power of the polynomial.
- Polynom derivative () const

  Takes the derivative of the polynomial.
- long long valueAtPoint (long long x) const

  Calculates the value of the polynomial at a given point.
- void normalization ()

  Normalization of the polynomial.
- std::vector< Polynom > findRoots ()

  Finds roots of the polynomial.
- Polynom toThePower (long long pow) const

  Raises the polynomial to a given power.
- Polynom getWithOtherParameter (long long b) const

  Gets a polynomial of the form f(x-b)
- long long rootsNumber ()

  Implements an algorithm for finding number of roots for polynomial in accordance with Konig-Rados theorem.
- Polynom gcd (const Polynom &other)

  Calculates the greatest common divisor of two polynomials.
- std::vector< Polynom > factorizeCyclotomicRi (size_t n, size_t maxCount=0)

  Gets irreducible factors of nth cyclomotic polynomial using Ri polynomials.
- bool isIrreducible ()

  Checks if the polynomial is irreducible.
- Matrix buildBerlekampMatrix () const

  Finds Berlekamp matrix.
- std::string berlekampAlgorithm () const

  Factorizes the polynomial via Berlekamp algorithm.

Static Public Member Functions

- static Polynom cyclotomicPolynomial (int prime, int n)

  Calculates n-th cyclotomic polynomial.
- static std::vector< Polynom > allIrreduciblePolynomials (long long prime, long long n)

  Finds all irreducible polynomials of degree n.
- static std::vector< Polynom > nIrreduciblePolynomials (long long prime, long long n, int size)

  Finds "size" irreducible polynomials of degree n.
- static Polynom findIrreduciblePolynomial (long long prime, long long n)

  Finds one irreducible polynomial of degree n.

Friends

- Polynom operator+ (Polynom const &p1, Polynom const &p2)

  Adds polynomials over the field.
- Polynom operator- (Polynom const &p1, Polynom const &p2)

  Finds difference of polynomials over the field.
- Polynom operator∗ (Polynom const &p1, Polynom const &p2)

  Multiplies polynomials over the field.
- Polynom operator∗ (Polynom const &p, long long const &number)

  Multiplies a polynomial on an integer constant over the field.
- Polynom operator/ (Polynom const &p1, Polynom const &p2)

  Divides polynomials over the field.
- Polynom operator% (Polynom const &p1, Polynom const &p2)

  Takes the remainder of a division of polynomials over the field.

3.2.1 Detailed Description

class Polynom that describes polynomial in a ring GF(p)[X]

polynomials over field GF(p) (p is prime) coefficient by (mod p) power of the polynomial - n Every object consists of terms (PolyTerm) Every structure "PolyTerm" has fields "key" and "next" key - coefficient of term next - pointer to next object of structure "PolyTerm"

3.2.2 Member Function Documentation

3.2.2.1 allIrreduciblePolynomials()

```
std::vector< Polynom > Polynom::allIrreduciblePolynomials (
            long long prime,
            long long n )  [static]
```

Finds all irreducible polynomials of degree n.

#12

Author

Vladyslav Prokopchuk

Returns

Vector of all irreducible polynomials of degree n

3.2.2.2 berlekampAlgorithm()

```
std::string Polynom::berlekampAlgorithm ( ) const
```

Factorizes the polynomial via Berlekamp algorithm.

Author

Medynskyi Mykola, Pashchenko Dmytro

Returns

String representation of the factorized polynomial

3.2.2.3 buildBerlekampMatrix()

```
Matrix Polynom::buildBerlekampMatrix ( ) const
```

Finds Berlekamp matrix.

Author

Medynskyi Mykola, Pashchenko Dmytro

Returns

Berlekamp matrix

3.2.2.4 cyclotomicPolynomial()

```
Polynom Polynom::cyclotomicPolynomial (
            int prime,
            int n )  [static]
```

Calculates n-th cyclotomic polynomial.

#9

Author

Rostyslav Mochulskyi

3.2.2.5 derivative()

```
Polynom Polynom::derivative ( ) const
```

Takes the derivative of the polynomial.

#2

Author

Darik Ivashyn

3.2.2.6 factorizeCyclotomicRi()

```
std::vector< Polynom > Polynom::factorizeCyclotomicRi (
            size_t n,
            size_t maxCount = 0 )
```

Gets irreducible factors of nth cyclomotic polynomial using Ri polynomials.

#10

Author

Hryshchenko Yurii

Parameters

amount	If maxCount > 0, finds "maxCount" factors at most, otherwise finds all factors

3.2.2.7 findIrreduciblePolynomial()

```
Polynom Polynom::findIrreduciblePolynomial (
            long long prime,
            long long n )  [static]
```

Finds one irreducible polynomial of degree n.

#12

Author

Vladyslav Prokopchuk

Returns

Irreducible polynomial of degree n

3.2.2.8 findRoots()

```
std::vector< Polynom > Polynom::findRoots ( )
```

Finds roots of the polynomial.

#3

Author

Karina Masol & Yuriy Momotenko

3.2.2.9 gcd()

```
Polynom Polynom::gcd (
            const Polynom & other )
```

Calculates the greatest common divisor of two polynomials.

#7

Author

Nikita Pupov

3.2.2.10 getWithOtherParameter()

```
Polynom Polynom::getWithOtherParameter (
            long long b ) const
```

Gets a polynomial of the form f(x-b)

#3

Author

Karina Masol & Yuriy Momotenko

3.2.2.11 isIrreducible()

```
bool Polynom::isIrreducible ( )
```

Checks if the polynomial is irreducible.

#13

Author

Totskyi Alexander

Returns

1 - if irreducible, 0 - reducible

3.2.2.12 nIrreduciblePolynomials()

```
std::vector< Polynom > Polynom::nIrreduciblePolynomials (
            long long prime,
            long long n,
            int size )  [static]
```

Finds "size" irreducible polynomials of degree n.

#12

Returns

Vector of all irreducible polynomials of degree n

3.2.2.13 normalization()

```
void Polynom::normalization ( )
```

Normalization of the polynomial.

#2

Author

Darik Ivashyn

3.2.2.14 rootsNumber()

```
long long Polynom::rootsNumber ( )
```

Implements an algorithm for finding number of roots for polynomial in accordance with Konig-Rados theorem.

#4

Author

Yana Skyrda

3.2.2.15 toThePower()

```
Polynom Polynom::toThePower (
            long long pow ) const
```

Raises the polynomial to a given power.

#3

Author

Karina Masol & Yuriy Momotenko

3.2.2.16 valueAtPoint()

```
long long Polynom::valueAtPoint (
            long long x ) const
```

Calculates the value of the polynomial at a given point.

#2

Author

Darik Ivashyn

3.2.3 Friends And Related Function Documentation

3.2.3.1 operator%

```
Polynom operator% (
            Polynom const & p1,
            Polynom const & p2 )  [friend]
```

Takes the remainder of a division of polynomials over the field.

#6

Author

Daryna Bondarets

3.2.3.2 operator* [1/2]

```
Polynom operator* (
            Polynom const & p,
            long long const & number )  [friend]
```

Multiplies a polynomial on an integer constant over the field.

#1

Author

Daryna Bondarets

3.2.3.3 operator* [2/2]

```
Polynom operator* (
            Polynom const & p1,
            Polynom const & p2 )  [friend]
```

Multiplies polynomials over the field.

#1

Author

Daryna Bondarets

3.2.3.4 operator+

```
Polynom operator+ (
            Polynom const & p1,
            Polynom const & p2 )  [friend]
```

Adds polynomials over the field.

#1

Author

Daryna Bondarets

3.2.3.5 operator-

```
Polynom operator- (
            Polynom const & p1,
            Polynom const & p2 )  [friend]
```

Finds difference of polynomials over the field.

#1

Author

Daryna Bondarets

3.2.3.6 operator/

```
Polynom operator/ (
            Polynom const & p1,
            Polynom const & p2 )  [friend]
```

Divides polynomials over the field.

#6

Author

Daryna Bondarets

The documentation for this class was generated from the following files:

- Algebra-polynom/Polynom/Polynom.h
- Algebra-polynom/Polynom/Polynom.cpp

Chapter 4

File Documentation

4.1 Algebra-polynom/Polynom/GaloisField.h File Reference

Header file for class GaloisField.

```
#include "Polynom.h"
```

Classes

- class GaloisField

  class GaloisField that describes finite field of polynomials generated by irreducible polynomial

4.1.1 Detailed Description

Header file for class GaloisField.

Definition of the class GaloisField

4.2 GaloisField.h

Go to the documentation of this file.

```
1
7 #include "Polynom.h"
8
17 class GaloisField {
18 private:
19     // irreducible polynomial that defines the field
20     Polynom irreducible;
21     // prime number that defines field GF(p) of coeffcients
22     long long prime;
23     // power of irreducible polynomial
24     long long degree;
25
26 public:
27     GaloisField() :prime(2), degree(2) {
28         irreducible = Polynom::findIrreduciblePolynomial(2, 2);
29     }
30
31     GaloisField(long long prime, long long degree) :prime(prime), degree(degree){
32         irreducible = Polynom::findIrreduciblePolynomial(prime, degree);
```

```
33      }
34      long long getPrime() const { return prime; }
35      long long getDegree() const { return degree; }
36      Polynom getIrreducible() const {
37          return irreducible;
38      }
39
40      std::vector<Polynom> getNIrreducible(int n) const {
41          return Polynom::nIrreduciblePolynomials(prime, degree, n);
42      }
43
44      void setIrreducible(int n, int index) {
45          std::vector<Polynom> temp = getNIrreducible(n);
46          irreducible = temp[index % n];
47      }
48
49      void setIrreducible(Polynom p) {
50          irreducible = p;
51          degree = p.getPolyPower();
52      }
53
57      Polynom add(Polynom const& left, Polynom const& right) const {
58          if (left.getPrime() != right.getPrime() ||
59              left.getPrime() != irreducible.getPrime()) return Polynom();
60          return (left + right) % irreducible;
61      }
62
66      Polynom subtract(Polynom const& left, Polynom const& right) const {
67          if (left.getPrime() != right.getPrime() ||
68              left.getPrime() != irreducible.getPrime()) return Polynom();
69          return (left - right) % irreducible;
70      }
71
75      Polynom multiply(Polynom const& left, Polynom const& right) const {
76          if (left.getPrime() != right.getPrime() ||
77              left.getPrime() != irreducible.getPrime()) return Polynom();
78          return (left * right) % irreducible;
79      }
80
84      Polynom extendedEuclideanAlgorithm(Polynom a, Polynom b, Polynom *x, Polynom *y)const {
85          long long prime=a.getPrime();
86
87          if (a == Polynom(prime,std::vector<long long>{0}))
88          {
89              *x = Polynom(prime,std::vector<long long>{0});
90              *y = Polynom(prime,std::vector<long long>{1});
91              return b;
92          }
93          Polynom x1, y1;
94          Polynom d = extendedEuclideanAlgorithm(b%a, a, &x1, &y1);
95          *x = y1 - (b / a) * x1;
96          *y = x1;
97          return d;
98      }
99
103      Polynom getInverse(Polynom const& polynom) const {
104          Polynom x;
105          Polynom y;
106          extendedEuclideanAlgorithm(polynom, irreducible, &x, &y);
107          x = (x % irreducible + irreducible) % irreducible;
108          return x;
109      }
110
114      Polynom divide(Polynom const& left, Polynom const& right) const {
115          if (left.getPrime() != right.getPrime() ||
116              left.getPrime() != irreducible.getPrime()) return Polynom();
117          return (left * getInverse(right)) % irreducible;
118      }
119
123      Polynom mod(Polynom const& left, Polynom const& right) const {
124          if (left.getPrime() != right.getPrime() ||
125              left.getPrime() != irreducible.getPrime()) return Polynom();
126          return (left % right) % irreducible;
127      }
128
132      Polynom derivative(Polynom const& polynom) const {
133          return polynom.derivative() % irreducible;
134      }
135  };
```

4.3 Algebra-polynom/Polynom/Polynom.cpp File Reference

Implementation of class Polynom.

```
#include "Polynom.h"
#include "../utils.h"
#include <list>
#include <stack>
```

Functions

- Polynom operator+ (Polynom const &pol1, Polynom const &pol2)
- Polynom operator- (Polynom const &pol1, Polynom const &pol2)
- Polynom operator∗ (Polynom const &pol1, Polynom const &pol2)
- Polynom operator∗ (Polynom const &p, long long const &number)
- Polynom operator/ (Polynom const &p1, Polynom const &p2)
- Polynom operator% (Polynom const &p1, Polynom const &p2)

4.3.1 Detailed Description

Implementation of class Polynom.

Functional of polynomial

4.3.2 Function Documentation

4.3.2.1 operator%()

```
Polynom operator% (
            Polynom const & p1,
            Polynom const & p2 )
```

#6

Author

Daryna Bondarets

4.3.2.2 operator∗() [1/2]

```
Polynom operator* (
            Polynom const & p,
            long long const & number )
```

#1

Author

Daryna Bondarets

4.3.2.3 operator*() [2/2]

```
Polynom operator* (
            Polynom const & pol1,
            Polynom const & pol2 )
```

#1

Author

Daryna Bondarets

4.3.2.4 operator+()

```
Polynom operator+ (
            Polynom const & pol1,
            Polynom const & pol2 )
```

#1

Author

Daryna Bondarets

4.3.2.5 operator-()

```
Polynom operator- (
            Polynom const & pol1,
            Polynom const & pol2 )
```

#1

Author

Daryna Bondarets

4.3.2.6 operator/()

```
Polynom operator/ (
            Polynom const & p1,
            Polynom const & p2 )
```

#6

Author

Daryna Bondarets

4.4 Algebra-polynom/Polynom/Polynom.h File Reference

Header file for class Polynom.

```
#include <iostream>
#include <cmath>
#include <vector>
#include <string>
#include <algorithm>
#include "../Matrix/Matrix.h"
```

Classes

- class Polynom

  class Polynom that describes polynomial in a ring GF(p)[X]

4.4.1 Detailed Description

Header file for class Polynom.

Definition of the class Polynom

4.5 Polynom.h

Go to the documentation of this file.

```
1
7 #pragma once
8
9 #include <iostream>
10 #include <cmath>
11 #include <vector>
12 #include <string>
13 #include <algorithm>
14 #include "../Matrix/Matrix.h"
15
27 class Polynom
28 {
29 private:
30     long long prime; // p
31     // Structure that describes a single term of the polynomial
32     struct PolyTerm
33     {
34         long long key;  // Coefficient of the term
35         long long pow;  // Power of the term
36         PolyTerm *next; // Pointer to the next term
37
38         /*destructor*/
39         ~PolyTerm()
40         {
41             if (next)
42                 delete next;
43         }
44     };
45     PolyTerm *head; // Pointer to the first term of polynom
46
47 public:
48     /*constructors*/
49     Polynom();
50     Polynom(long long _prime, std::vector<long long> keys);              //for all terms
51     Polynom(long long _prime, std::vector<std::vector<long long» keys); //for some terms
52     Polynom(long long _prime, std::string polynom, char X);
53     Polynom(const Polynom &other)
54     { // copy constructor
```

```
55          this->prime = other.prime;
56          head = nullptr;
57          PolyTerm *tmp = other.getHead();
58          while (tmp)
59          {
60              addItem(makeItem(tmp->pow, tmp->key));
61              tmp = tmp->next;
62          }
63          tmp = nullptr;
64      }
65      Polynom(Polynom &&other) noexcept
66      { // move constructor
67          this->prime = other.prime;
68          this->head = other.head;
69          other.head = nullptr;
70      }
71      /*destructor*/
72      ~Polynom()
73      {
74          if (head)
75              delete head;
76      }
77
78      /*Getters and Setters*/
79      PolyTerm *getHead() const { return head; }
80      PolyTerm *getTerm(long long pow) const
81      { //retuns term by its power
82          PolyTerm *tmp = head;
83          if (head == nullptr)
84              return nullptr;
85          while (tmp)
86          {
87              if (tmp->pow == pow)
88                  return tmp;
89              tmp = tmp->next;
90          }
91          return nullptr;
92      }
93      long long getTermKey(long long pow) const
94      { //retuns coef of the term by its power
95          PolyTerm *term = getTerm(pow);
96          if (term)
97              return term->key;
98          return 0;
99      }
100      // returns power of the field
101      long long getPrime() const { return prime; }
102      void setHead(PolyTerm *_head) { head = _head; }
103      void setPrime(long long _prime) { prime = _prime; }
104      void operator=(const Polynom &other)
105      {
106          prime = other.getPrime();
107          if (head)
108              delete head;
109          head = nullptr;
110          PolyTerm *tmp = other.getHead();
111          while (tmp)
112          {
113              addItem(makeItem(tmp->pow, tmp->key));
114              tmp = tmp->next;
115          }
116          tmp = nullptr;
117      };
118
122      std::string show() const;
123
127      long long getPolyPower() const
128      {
129          long long pow = 0;
130          PolyTerm *tmp = head;
131          if (head == nullptr)
132              return pow;
133          while (tmp)
134          {
135              if (tmp->pow > pow)
136              {
137                  pow = tmp->pow;
138              }
139              tmp = tmp->next;
140          }
141          return pow;
142      }
143
144      // Creates new term (PolyTerm) with coefficient val
145      PolyTerm *makeItem(long long pow, long long val);
146      // Adding term to the polynomial in non-descending order
147      void addItem(PolyTerm *el);
```

```
148
153     friend Polynom operator+(Polynom const &p1, Polynom const &p2);
154
159     friend Polynom operator-(Polynom const &p1, Polynom const &p2);
160
165     friend Polynom operator*(Polynom const &p1, Polynom const &p2);
166
171     friend Polynom operator*(Polynom const &p, long long const &number);
172     friend Polynom operator*(long long const &number, Polynom const &p);
173
178     friend Polynom operator/(Polynom const &p1, Polynom const &p2);
179
184     friend Polynom operator%(Polynom const &p1, Polynom const &p2);
185
190     Polynom derivative() const;
191
196     long long valueAtPoint(long long x) const;
197
202     void normalization();
203
208     std::vector<Polynom> findRoots();
209
214     Polynom toThePower(long long pow) const;
215
220     Polynom getWithOtherParameter(long long b) const;
221
227     long long rootsNumber();
228
233     Polynom gcd(const Polynom& other);
234
235     /* #9
236      * @author Nikita Pupov
237      * @brief Equality operator
238      */
239     friend bool operator==(Polynom const &p1, Polynom const &p2);
240
245     static Polynom cyclotomicPolynomial(int prime, int n);
246
252     std::vector<Polynom> factorizeCyclotomicRi(size_t n, size_t maxCount = 0);
253
259     static std::vector<Polynom> allIrreduciblePolynomials(long long prime, long long n);
260
265     static std::vector<Polynom> nIrreduciblePolynomials(long long prime, long long n, int size);
266
272     static Polynom findIrreduciblePolynomial(long long prime, long long n);
273
279     bool isIrreducible();
280
285     Matrix buildBerlekampMatrix() const;
286
291     std::string berlekampAlgorithm() const;
292
293 protected:
294     /* #1
295      * @brief Adds polynomials over the field
296      */
297     Polynom addPoly(Polynom const &p1, Polynom const &p2);
298
299     /* #1
300      * @brief Finds difference of polynomials over the field
301      */
302     Polynom diffPoly(Polynom const &p1, Polynom const &p2);
303
304     /* #1
305      * @brief Multiplies polynomials over the field
306      */
307     Polynom multPoly(Polynom const &p1, Polynom const &p2);
308
309     /* #1
310      * @brief Multiplies polynomials over the field with power=1
311      */
312     Polynom multSimple(Polynom const &p1, Polynom const &p2);
313
314     /* #1
315      * @brief Multiplies polynomial on an integer constant over the field
316      */
317     Polynom multNumber(Polynom const &p, long long const &number);
318
319     /* #6
320      * @brief Multiplies polynomials over the field
321      */
322     Polynom multPolyforDivide(Polynom const &p1, Polynom const &p2);
323
324     /* #6
325      * @brief Divides polynomials over the field
326      */
327     std::pair<Polynom, Polynom> simple_division(Polynom const &p1, Polynom const &p2) const;
```

```
328
329     /* @author  Datsiuk Vitaliy, Medynskyi Mykola
330      * @brief Computes the square free decomposition of the given polynomial
331      * list of pairs. The first element of the pair is a factor of the given
332      * polynomial, and the second, its multiplicity.
333      * @return std:vector of pairs<Polynom, int>
334      */
335     std::vector<std::pair<Polynom, long long» squareFreeDecomposition() const;
336
337     /* @author Medynskyi Mykola
338      * @brief Finds pth root of a given polynomial
339      * @return Polynomial
340      */
341     Polynom pthRoot(Polynom f);
342
343     std::vector<std::pair<std::vector<Polynom>, long long»
      berlekampAlgorithmMainCase(std::vector<std::pair<Polynom, long long» const& unmultiple_factors)
      const;
344
345     /* @author Pashchenko Dmytro
346      * @brief Builds polynomial basis of solution space of comparison system h^p = h (mod f)
347      * @return Basis polynomials
348      */
349     std::vector<Polynom> getComparisonSystemSolutionBasis() const;
350
351     std::vector<std::pair<Polynom, long long»
      sort_polynomials_by_power(std::vector<std::pair<std::vector<Polynom>, long long» const& polynomials)
      const;
352
353     /* @author Pashchenko Dmytro
354      * @brief Factorizes unmultiple factors using basis polynomials (f = ⋄(gcd(f, h - a)))
355      * @return std::vector of pairs "polynomials, their multiplicity"
356      */
357     std::vector<std::pair<std::vector<Polynom>, long long»
      factorizeByBasisPolynomials(std::vector<std::pair<Polynom, long long» const &unmultiple_factors,
358         std::vector<Polynom> const &basis) const;
359 };
```